freefloat

Freefloat Access

version 2.5

User's Manual

Contacting Freefloat 1
Introduction 2
System Requirements 2
Supported Windows Devices 2
Terminal Emulation 3
Product Features 3
Emulator Features 4
5250 Features 5
Getting Started 6
A Typical Installation 6
Step 1, testing the host connection from your PC 6
Step 2, configure and install on CE device 6
Step 3, licensing the emulator 7
The Emulator 10
Screen Layout 10
Emulator Text Area 11
Keyboard 12
Status Area 13
The Splash Screen 14
Splash Screen Mode 14
Locating the cursor, centering the screen around it 15
Scrolling the screen without moving the cursor position (moving the viewport) 15
Moving the cursor (insertion point) 15
Scanner 16
Scanner On/Off Control in 3270/5250 emulation 16
Scanner On/Off Control in ANSI emulation 16
Display Character Translation 17
Screen Modes 19
Standard Mode 19
Locked Mode 19
Center Cursor Mode 19
Screen Cursor Mode 20
Corner Mode 20
Full screen Mode 20
VT/ANSI 21
Line Mode 21
Attribute Mapping 22
Printing 23
Logging 24
The Configuration Application 25
The Screen Layout 25
Option Types 26
The CEInstaller Application 27

What is transferred during installation and where is it? 28
Keyboard Mapping 29
How to map a key 29
Key Table 30
Local Action Keys 34
Emulation Actions 35
Configurable Keyboard 35
Scripting 36
Overview 36
The Lua Language 37
Script Events 38
Event Methods 39
onKey(keyname) 39
onStart() 39
onEnd() 40
onActivate() 40
onDeactivate() 40
onCursorMove() 41
onEvent(eventname) 41
onNetworkData() 41
onMouseClick() 42
onMouseWheel(delta, shift, control) 42
Script Methods 43
Output 46
Text Buffer 52
Cursor 55
Scanner and Barcode 57
Viewport 60
Emulation Type 67
Time & Timer Events 68
Session Control 75
Application Launch 78
Font 81
On-screen Keyboard Control 85
Freefloat Link Integration 94
Hotspots 97
Appendix A – Model Specifics 98
Casio DT-X7 98
Casio DT-X8 100
Denso BHT-400 102
Symbol PPT8846 102
Machines that have a sound device capable of playing wav-files 102
PSC Falcon 44xx with monochrome screen 102
Appendix B – Logging Network Data 104

Appendix C – EBCDIC Character Tables 105
Code Page 278, Swedish/Finnish 105
Code Page 37, USA/Canada 106
Appendix D – TN5250 Auto-signon 107
Appendix E – TN5250 Enhancement Settings 107
Appendix F – Command Line Parameters 108
Appendix G – Version History 109

Copyright © 2003–2012 Freefloat AB

Contacting Freefloat

Forum	http://forum.freefloat.com/
Web	http://www.freefloat.com/
E-mail	mailto:info@freefloat.com

Introduction

Freefloat Access was developed to enable the many users of IBM mainframes and various *NIX and other ANSI/VT-compatible systems to seamlessly integrate these devices into their daily operations at a reasonable cost.

The emulator supports VT52, VT100 and VT220, TN3270, and TN5250.

Functions that can't be supported on certain devices due to hardware limitations are either ignored or mapped as appropriate.

System Requirements

Note! The below requirements are for the PC, onto which the product is installed. The actual terminal emulator runs on the devices listed below.

- Windows XP/2000/Vista
- Minimum 16 MB memory, (the Windows emulator takes max 3 MB of memory)
- The disk space required is displayed by the installation program
- Microsoft ActiveSync v3.5 or later (for installing on a CE device, version 3.5, 3.7, 3.7.1, and 3.8.0 has been tested)
- Adobe Acrobat Reader 4.05 or later

Supported Windows Devices

For a list of supported devices, please refer to the product page for Freefloat Access on www.freefloat.com.

Terminal Emulation

- IBM 3278-2 (24 x 80 monochrome)
- IBM 3279-2 (24 x 80 color)
- IBM 5251-11 (24 x 80 monochrome)
- IBM 3179-2 (24 x 80 color)
- DEC VT52, VT100, VT220

Product Features

- Configuration application
- Installation program for installing the emulator and configuration on Windows CE devices
- Freefloat Key integrated, configurable layouts via Freefloat Key Designer
- Full screen emulator for demonstration and testing purposes enabling you to test the configuration, scripts, etc. before deploying onto the CE device
- Host simulators for VT, 3270, and 5250 which enables you to test the emulator on the CE device without having access to a real host

Emulator Features

- **NEW!** The dynamic script language Lua has been embedded which makes Freefloat Access very powerful, flexible and extensible
- The ability to block the user from accessing the Windows CE shell
- Support for multi-session to the same or a different server with the same or different emulation type
- Variable font size, bold or normal
- Large user-configurable keyboard
- Solid or blinking cursor
- Hotspot support
- Color support in all emulations
- Fully customizable colors
- VT/ANSI attributes can be mapped to colors
- VT/ANSI big text (double width, double width/double height)
- Supported text attributes: Underline, Reverse, Intensify (Bold), Blink, Colors, Invisible, Column Separator (5250)
- Centering of the viewport on cursor and/or movement keys
- Manual or automatic centering of the viewport
- Viewport locking
- Fully customizable character translation between the host and the device
- Customizable key strings
- Real full-screen usage on CE device, even hides (optionally) the taskbar.
- Tight device integration (for example barcode reader, radio control, beeper & vibrator)
- Customizable mouse (touch screen) behavior
- A different kind of signal (beep and/or vibrator) controllable from the host
- VT/ANSI pass-through printing

5250 Features

- Handles more than 256 fields on one screen correctly.
- Supports the 5250 Text Window definition with full attribute support, header, footer including attributes and orientation for header and footer.
- 100 % color and attribute support in both color and monochrome mode.
- Handles multi-line fields correctly, even fields that cover the entire screen.
- Full and correct field type/field properties support:

  Alphabetic Only (also accepts international letters according to IBM's specifications)
  Alphabetic Shift (actually means all characters are allowed)
  Monocase (also uppercases international letters)
  Signed Numeric
  Numeric Only
  Bypass
  Field Exit Required
  Mandatory Entry
  Mandatory Fill
  Right Adjust
  Zero Fill
  Space Fill
  Dup Enable
  Numeric Shift (actually the same as Alphabetic Shift)
  Modified (the host can set the modified status of a field directly in a screen definition)

Getting Started

Since you are reading this you have managed to install the product on a PC. A program group called **Freefloat Access** was created on your Start menu.

The program group contains icons for configuring the various emulators (CE device specific ones and the one for regular Windows), the CEInstaller program which installs the emulator on a CE device as well as the configuration, support files, character mapping files, and optionally scripts.

There is also an icon for starting the emulator on your PC.

A Typical Installation

Let's say you have a 3270 host. It's IP address is 192.168.1.3. It serves TN3270 connections on port 23. You also have a Denso CE device that you want to connect to the host.

Step 1, testing the host connection from your PC

1. Select the icon **Config Freefloat Access** on the Start menu.
2. The configuration application is displayed, the configuration for the regular Windows version of the emulator is automatically opened.
3. In the configuration application, go to the option *Common\Connection\Host*.
4. Enter the IP address for your host (192.168.1.3).
5. Go to the option *Common\Connection\Emulation Type*
6. Select the emulation (TN3270 Color 24x80).
7. Choose File/Save to save the modified configuration.
8. Exit the configuration application.
9. Start Freefloat Access by selecting the icon **Freefloat Access** on the Start menu.
10. Make sure you can connect to your host.

This must be done with a new CE device prior to installing the emulator on it:

> *Start your CE device and configure your CE device's RF network card so it can associate with an access point. For example: SSID (network name), IP address (fixed or dynamic), WEP encryption keys or other encryption settings if applicable. Please see the documentation for the CE device on how to do this.*

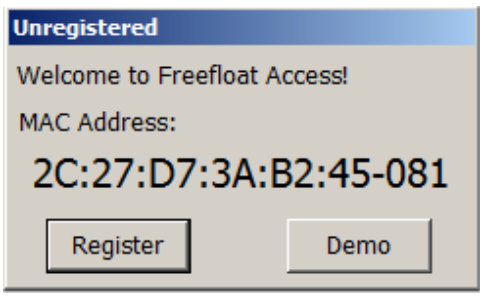
Step 2, configure and install on CE device

1. Select the icon **Config <manufacturer name>/<device name>** depending on what CE device you are using.
2. Go to the *Host* option and enter your host's IP address.
3. Go to the option *Emulation Type* and select the desired emulation.

4. Save the configuration and click on in the toolbar of the configuration application. This will start the **CEInstaller** application. It installs the emulator on the CE device.
5. Start the CE device and make sure it is synchronized with your PC via Microsoft ActiveSync.
6. Click on the **Install** button.
7. After the emulator is installed, start it and check that the emulator connects to your host.

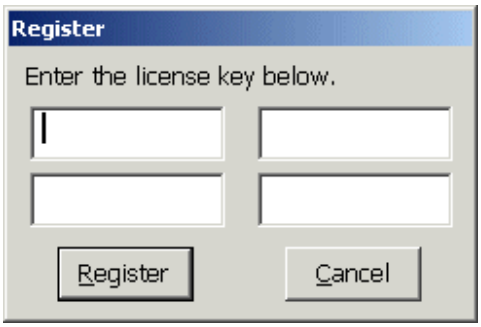
Step 3, licensing the emulator

When the emulator is started on the CE device for the first time it is unlicensed. The following dialog box is displayed:

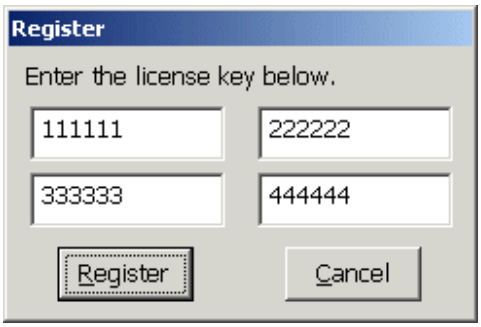


To license the emulator, write down the MAC address, contact your reseller or Freefloat to buy a license. You will then receive a license key. Until you have received the license key, you can run the emulator in demonstration mode. The demonstration mode has no limitations except that it times out and exits after 20 minutes.

When you receive the license key, click on the Register button in the dialog box above. The Register dialog box will be displayed:

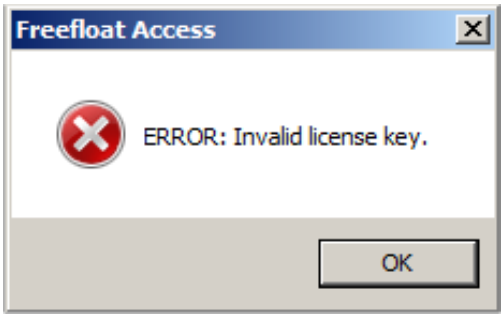


Enter the digits from the license key into the boxes. Leave out the hyphens (-). If the license key you received was 111111-222222-333333-444444 your dialog should look like this:



Click on the **Register** button to complete the registration process.

If you made a mistake when entering the license key, you will get the following message when you click the register button:



When you click the OK button you will be taken back to the Register dialog box.

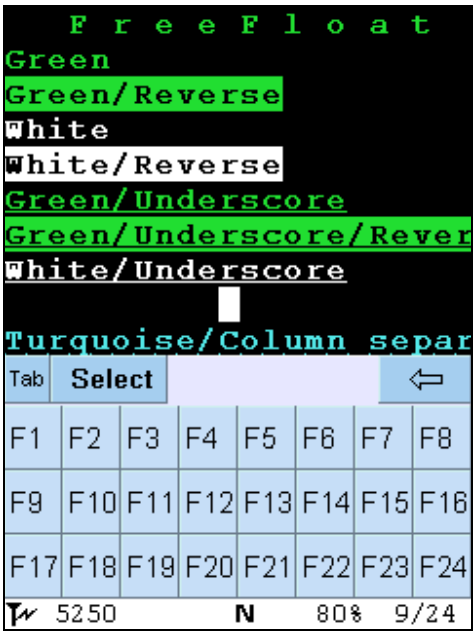
The Emulator

This topic explains the different screen objects present in the emulator.

Please note that the window size of a CE device can be simulated in the regular Windows version of the emulator by setting the *Win32\Window Size* option to *Quarter VGA*.

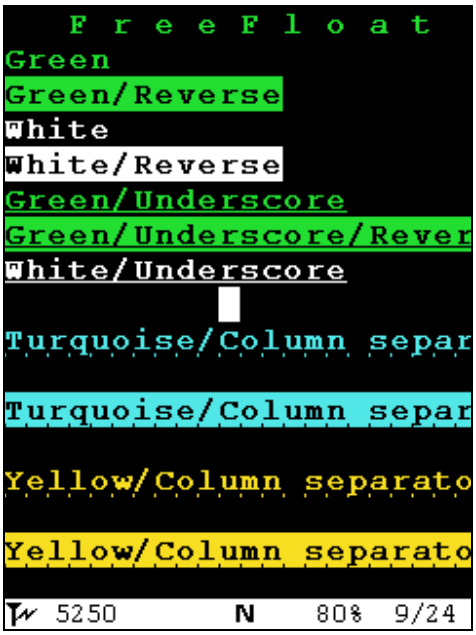
Screen Layout

Here's how the emulator might look like on a Quarter VGA color device:



Emulator Text Area

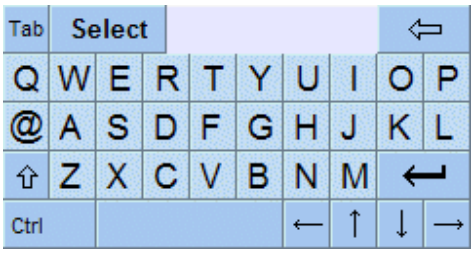
The emulator text area is the area above the status area.



Data from the host is in this area. Please note that the keyboard covers the text area when visible. When you touch or click on the emulator text area the action depends on the **onMouseClick** event in the script.

Keyboard

The keyboard is displayed below the emulator text area:



There are canvases for characters (even international ones), function keys, movement keys, and some special control buttons.

If you click on the canvas background, i.e. not on a button, you can drag the canvas to a new position.

Use Select to select another canvas.

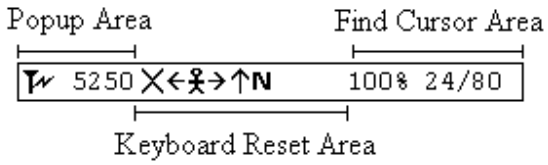
Exit will end the host session.

Font + Font - increases and decreases the font size used in the emulator text area.

Note! Changes to the font size is not saved to the configuration file. To make permanent changes to any option you need to use the configuration application and then transfer the new configuration file to the CE device.

About displays the version of the emulator.

Status Area



- is the radio connection indicator.

This indicator only works on Denso and Motorola/Symbol handhelds (and some other models), on other machines it displays as ( ).
If the flash disappears ( ) the CE device has lost its connection with the Access Point.

5250 this area displays the current emulation, ANSI, 3270 or 5250. In VT/ANSI emulation this text can be _ANSI, LANSI or lANSI. If a space, the emulator is not in linemode, if L the emulator is in negotiated linemode, if l (lowercase L) it is forced local linemode.

Clicking on the *Popup Area* toggles the keyboard.

is the 3270/5250 operator error indicator.

	You tried to enter text outside a field
NUM	You tried to enter invalid characters into a numeric only field
	You tried to enter more data into a field than there is room for
SYS	The emulator is waiting for a host response
	Data has been received from the host and is being processed
↑	Insert mode indicator
A or **N**	Input Mode indicator (only active on some models) Tells you whether the input mode is alpha or numeric. On a Denso BHT-200 you press the ALP key to toggle.

If the keyboard is locked, indicated by , NUM, and , you can click on the *Keyboard Reset Area* to unlock the keyboard.

100% is the battery level.

24/80 is the cursor location indicator. To quickly locate the current cursor position (the active input), click on the *Find Cursor Area*.

The Splash Screen

When the connection to the host is terminated the Splash screen is displayed.

When the option *Forced Mode Password* has been given a value, a dialog box will appear. To be able to exit the emulator, the correct password must be entered.

The Splash screen is also used for displaying network error messages.

Here's how the Splash screen might look like:

Splash Screen Mode

There are two different splash screen modes, the normal one with Re-connect and Exit buttons and a more specialized mode where the Reboot button has been made very large and centered at the bottom of the splash screen. In this specialized mode there is also a Reboot button which reboots Windows. The Reboot mode for the splash screen is only useable in Windows 2000/XP, not in Windows CE. Also it need VGA resolution or higher. The option that controls the splash screen mode is:

Common\Screen\Splash Screen Mode

Please note that the following three functions will NOT move the viewport when the screen has been locked with the script command **app.lockWindow()**.

Locating the cursor, centering the screen around it

Due to the tiny screen of most CE devices, it is very easy to loose track of the cursor position. To quickly locate the current cursor position (the active input), the following shortcut method can be used.

Touch or click on the *Find Cursor Area*.

The emulator will try to center the viewport around the cursor, so the operator can see as much as possible of the screen at that position. This doesn't move the cursor, but moves the viewport to the cursor position.

Scrolling the screen without moving the cursor position (moving the viewport)

Push and drag in the emulator text area. Please note that the scrolling can be disabled by the script.

If you need viewport scrolling from the Freefloat Key keyboard, define four keys that send Alt+arrow keys.

Moving the cursor (insertion point)

If the option *Center On Arrows* is enabled, the emulator will try to center the screen around the cursor, so the operator can see as much as possible of the screen at that position. This moves both the cursor and the viewport to the new cursor position.

If the option *Center On Arrows* is disabled, only the cursor moves, so if the cursor moves off the visible screen, the operator must use the function Locating the cursor, centering the screen around it described above, to locate the cursor.

On the keyboard canvas with arrow keys, use the arrow keys to move the cursor to the screen position desired.

If the CE device has hardware cursor keys, they can also be used to move the cursor.

Scanner

This section only applies to models where Freefloat Access has direct control of the barcode device

You can disable the option *Barcode Reader* if you are using a third-party barcode wedge software.

If the option *Barcode Reader* is disabled, the barcode parameters will have no effect.

All scanned data is internally converted to keystrokes, so any data scanned can also be input from the keyboard.

Please note that you may customize what happens when a barcode is read by modifying the script code for the event **onBarcode**.

Scanner On/Off Control in 3270/5250 emulation

By default, Freefloat Access turns off the scanner when the cursor is outside any input field or when there are no input fields on the screen. But if for example you need to simulate AID keys (function keys) by reading a barcode, the scanner needs to be enabled at all times by enabling the parameter *Common\Barcode\Barcode Reader Always On*.

Scanner On/Off Control in ANSI emulation

CSI % 0 S disables the scanner.
CSI % 1 S enables the scanner.

Display Character Translation

General Description
As delivered, the emulator is configured for the Swedish character set in 3270 or 5250 mode (Codepage 278), and in ANSI mode it uses the DEC Multinational character set (ISO/IEC 8859-1 Latin 1).

ANSI Mode
ANSI characters from the wire are converted to Unicode using the ansi2uni.map table before being stored in the screen buffer for subsequent display. Characters entered via keyboard are translated using the uni2ansi.map table before they are put on the wire.

3270 or 5250 Mode
Input from the keyboard is converted to EBCDIC using the uni2ebc.map table before being put into the buffer. All characters in the buffer (whether they came from the host or from the keyboard) are converted from EBCDIC to Unicode using the ebc2uni.map table before being displayed on the screen.

You can change the default character mapping via one of the four mapping files:

Filename	Mapping function
uni2ebc.map	Unicode (from keyboard) to EBCDIC (in buffer) mapping.
ebc2uni.map	EBCDIC (in buffer) to Unicode (screen) mapping.
uni2ansi.map	Unicode to ANSI (VT220 Multinational) mapping.
ansi2uni.map	ANSI (VT220 Multinational) to Unicode mapping.

If any of these files exist, it will be used to change the default character mapping for the characters defined in that file. Only the characters defined in that file are changed from it's default mapping.

Sample contents of ansi2uni.map:

```
// ANSI2UNI translates from network ISO11 Swedish (7-bit ASCII)
// to screen Unicode
//
// ISO11 Swedish value arrived from the wire, Unicode value
// to display on screen

0x60,0xe9
0x7b,0xe4
0x7c,0xf6
0x7d,0xe5
0x7e,0xfc
0x5b,0xc4
0x5c,0xd6
0x5d,0xc5
0x40,0xc9
0x5e,0xdc
```

The corresponding file uni2ansi.map would normally reverse this mapping as shown below:

```
// UNI2ANSI translate from keyboard Unicode to network ISO11 Swedish
//
// Unicode value arrived from the keyboard, ISO11 Swedish value
// to send to host

0xe9,0x60
0xe4,0x7b
0xf6,0x7c
0xe5,0x7d
0xfc,0x7e
0xc4,0x5b
0xd6,0x5c
0xc5,0x5d
0xc9,0x40
0xdc,0x5e
```

Screen Modes

The screen modes govern which portion of the larger terminal screen the emulator initially displays and how this viewport moves as the cursor moves.

Standard Mode

This is the default mode and it begins with the window in the upper left corner of the larger terminal.

The user is free to move the viewport, either automatically as defined by the options *Center On Arrows*, *Center On Tab* and *Center On Backspace*, or by the two functions
Locating the cursor, centering the screen around it
Or
Scrolling the screen without moving the cursor position (moving the viewport)

Locked Mode

In locked mode the viewport is locked to the coordinates defined by the script command **app.setOrigin()**.

The default position is the upper left-hand corner of the display.

The viewport does not move around, and only the selected window area is visible.

Locked mode disables the windowing keys (arrow keys, tab, backtab and backspace) and only allows you to window around the physical display by the script command **app.setOrigin()**.

Center Cursor Mode

Center cursor mode works best for applications that use the entire 80-character by 24-line logical display.

In this mode, the cursor remains in the center of the terminal's screen if possible.

As the cursor moves within the logical display, the emulator's viewport window moves to keep the cursor as centered as possible, showing as much as possible of the logical display on all sides of the cursor position.

Screen Cursor Mode

Screen center cursor mode ensures that the cursor is always visible.

As the cursor moves within the logical display, the emulator's viewport window moves in increments of the viewport window's width and height. If, for example, the viewport can display 12 lines with 20 characters wide the screen will effectively be divided into 4x2 screen parts. The first part will contain column 1 to 20 of lines 1 to 12 and the last part will contain column 61 to 80 of lines 13 to 24.

When the cursor is moved (either by user or host) into one of these parts, the viewport will snap into the part.

To enable this mode set the parameter *Follow Cursor* to the value **Screen**.

Corner Mode

Corner mode begins with the window in the upper left corner of the larger terminal.
It tries to keep the cursor in the lower right corner of the display, showing as much as possible of the logical display above and to the left of the cursor position.
Corner mode works best for applications that use the upper left corner of the logical screen.

As the cursor moves off the right or bottom edge of the radio terminal's display, the window moves to show the cursor. The distance moved and when movement should occur is determined by the options *Row Offset* and *Column Offset*.

Full screen Mode

Regular Windows version only.
This mode use the entire screen, hides the taskbar.
Keys that bring up the Start menu or switch to other application are filtered, making it impossible to switch away from the emulator to any other program.

VT/ANSI

Line Mode

Do not activate this parameter for 3270 or 5250 emulation. 3270 and 5250 are both block-devices, where the whole screen is one big “line mode” and only transmits when Enter or a PF-key is pressed.

To activate ANSI Line mode, enable the option *Force Line Mode*.

In the default “character-at-a-time” mode, (also known as full duplex) every key press or character scanned from a barcode generates a network packet transmission to the host. The host then echoes the character back to the handheld, possibly pre-pending it with cursor positioning escape sequences and then the handheld displays the character on the screen. So, for every character pressed or scanned, the handheld generates one radio transmission and receives one or more packet(s).

For example, if we scan a 20-character barcode, the handheld will generate 20 packet transmissions, and receive at least 20 packets.

When the radio is active, the power consumption of the handheld is much higher, up to five times the normal consumption, so it is easy to see that if we can reduce the transmissions, the battery will last much longer.

Line mode will do just that.

As long as we remain on the current line, and the cursor position is to the right of the first input position, and no key is pressed that would move us away from the current line, (arrow keys, Page Up/Down, Enter or backspace beyond the first input position), every key pressed or character scanned is buffered in the handheld.

When a key is pressed that requires the collected data to be transmitted, all data is then sent as ONE packet.

Using the previous example, our 20 packet transmissions and receptions are reduced to one transmission, and usually also to one reception, a reduction by 20 times!
Naturally, the battery now lasts much longer.

Not all applications work correctly in linemode, but the majority can benefit from this option.

Since all echoing of characters are done locally, password entries will be visible, because the handheld always echoes, and has no knowledge about the contents format.

However, this can be beneficial, since it is very easy to enter the wrong key using the sometimes awkward keyboards present on some handheld's, but the tiny screen usually prevents anyone else but the operator to read what was entered.

Attribute Mapping

In VT/ANSI emulation, the attributes bold, reverse, underline, and blink can be mapped to a text color.

For example, to map the attribute bold to red, enable the parameter *VT/ANSI Emulation\Attribute Mapping\Map Bold*, and then make sure *VT/ANSI Emulation\Attribute Mapping\Bold Color* is set to red.

Mapped attributes have priorities in the order bold, reverse, underline, and blink. This means that if some text is both bold and underline and bold is mapped to red and underline to blue the text will be red because bold has higher priority than underline.

Printing

Supported modes are:
Page print
Prints all information currently displayed on terminal screen.

```
            Octal              Decimal            Hex
ESC[0i   033 133 060 151    027 091 048 105    1B 5B 30 69
```

Controller print
Print is passed through the terminal and displayed only on the printer.
The terminal will pass all codes except the print on and off commands to the printer.
The terminal will not interact with the operating system during print.

```
Print on:    Octal             Decimal             Hex
ESC[5i    033 133 065 151    027 091 053 105     1B 5B 35 69
```

Any data received after Print on is directed to the printer instead of to the screen.

```
Print off:   Octal             Decimal             Hex
ESC[4i    033 133 064 151    027 091 052 105     1B 5B 34 69
```

Configuration Settings

The setting *Port Name* can be any valid output device, the most common ones are LPTx: or COMx: .

If the port selected is a serial COM port, you must specify at least Baud, Parity, Data and Stop.

The settings *Port Settings* consist of the following (where each parameter is separated from the next by a blank/space):

Baud=*n* where *n* is any valid numeric baudrate.
Parity=*p* where *p* is **N** for None, **O** for Odd or **E** for Even parity.
Data= *n* where *n* is 5, 6, 7 or 8 databits.
Stop= *n* where *n* is 1 or 2.

Optional parameters are (default value if not specified shown as **bold**):

to=on/**off**	If on, printing will be aborted if no data could be printed for 60 seconds.
xon=on/**off**	Software handshake using the XON/XOFF protocol.
octs=on/**off**	Hardware handshake using the CTS line.
odsr=on/**off**	Hardware handshake using the DSR line.
dtr=**on**/off	Sets the DTR line (output) on/off.
rts=**on**/off	Sets the RTS line (output) on/off.
rts=tg	RTS is on only then transmitting.

Logging

The emulator maintains a log file where information about certain event(s) and error conditions are registered.

The start time and emulator version are always registered each time the emulator starts.

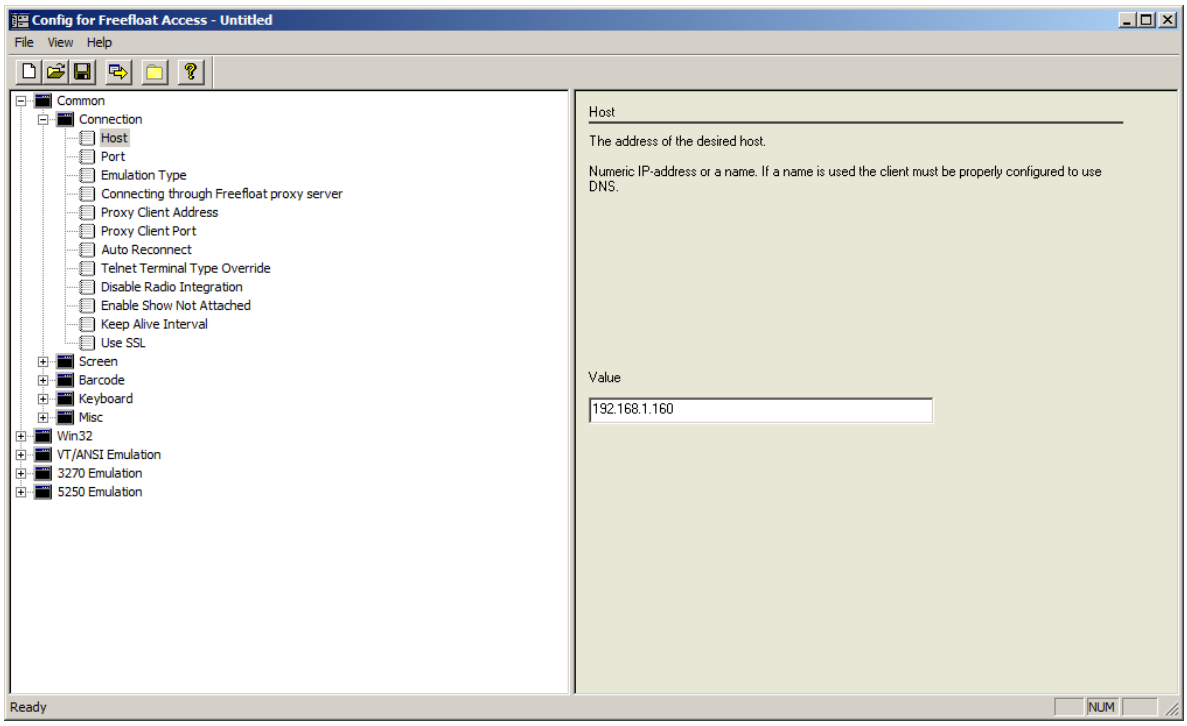
The log file is trimmed to the last 100 lines.

The name of log file is LOG.TXT and it's located in the root of the file system of the CE device. When running Freefloat Access on regular Windows, the log file is placed in same directory as the executable.

The Configuration Application

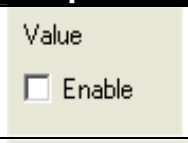
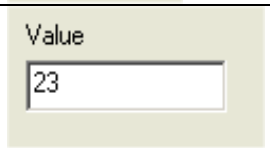
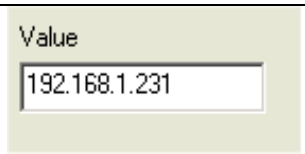
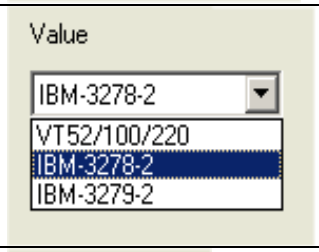
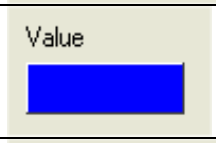
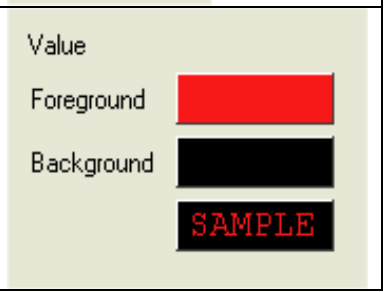
Use the Configuration Application to configure the emulator to your preferences. The emulator will only load it's settings from a configuration file called Terminal.cfg.

The Screen Layout



In the left window pane you select the option you want to change or inspect. When you click on an option the option's value is displayed in the right window pane together with a description of the option.

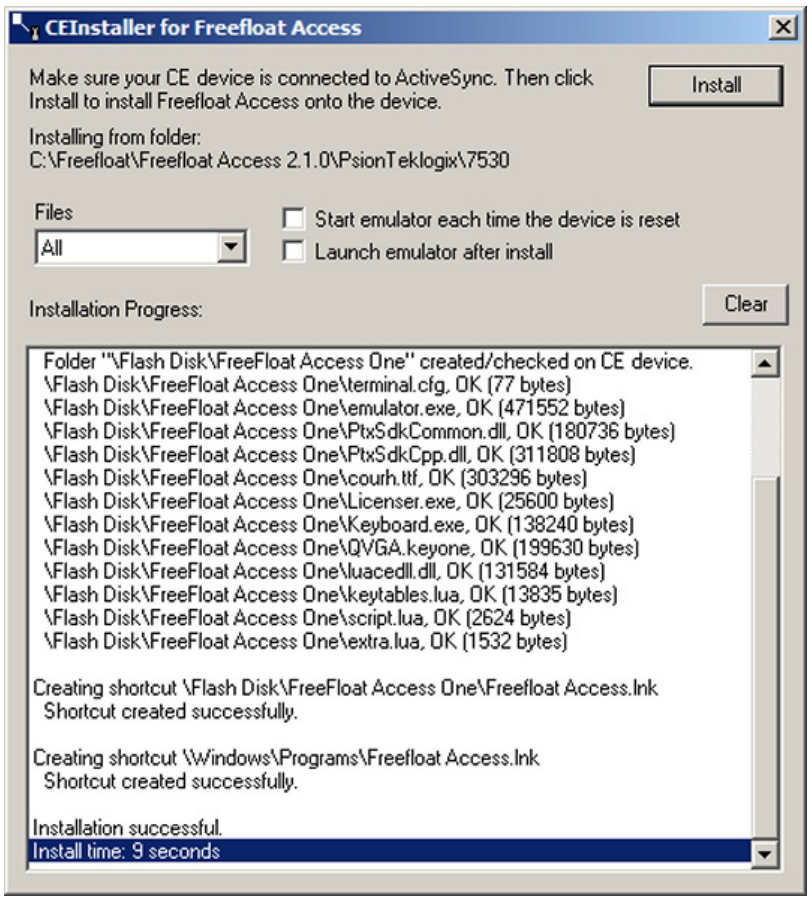
Option Types

Type	Description	Sample
Toggle	An option that can be enabled or disabled.	
Word	An option that holds an integer value. It also has a minimum and a maximum. If an invalid value is entered, a message will tell you that, together with the allowed range of values.	
String	An option that holds a text value.	
List	An option for which you can choose from a list of values.	
Color	A color option. Click on the button to select a color from the color palette dialog box.	
Color Pair	A color option that sets both foreground and background colors. There is also an area that display a sample with your color selection.	

The CEInstaller Application

CEInstaller is started when you click the install button .

It transfers the configuration, the emulator, some support files, and any scripts you might have written onto the CE device.


CEInstaller for Freefloat Access
Make sure your CE device is connected to ActiveSync. Then click Install to install Freefloat Access onto the device.
Install
Installing from folder:
C:\Freefloat\Freefloat Access 2.1.0\PsionTeklogix\7530
Files
All
Start emulator each time the device is reset
Launch emulator after install
Installation Progress:
Clear
Folder "\Flash Disk\FreeFloat Access One" created/checked on CE device.
\Flash Disk\FreeFloat Access One\terminal.cfg, OK (77 bytes)
\Flash Disk\FreeFloat Access One\emulator.exe, OK (471552 bytes)
\Flash Disk\FreeFloat Access One\PtxSdkCommon.dll, OK (180736 bytes)
\Flash Disk\FreeFloat Access One\PtxSdkCpp.dll, OK (311808 bytes)
\Flash Disk\FreeFloat Access One\courh.ttf, OK (303296 bytes)
\Flash Disk\FreeFloat Access One\Licenser.exe, OK (25600 bytes)
\Flash Disk\FreeFloat Access One\Keyboard.exe, OK (138240 bytes)
\Flash Disk\FreeFloat Access One\QVGA.keyone, OK (199630 bytes)
\Flash Disk\FreeFloat Access One\luacedll.dll, OK (131584 bytes)
\Flash Disk\FreeFloat Access One\keytables.lua, OK (13835 bytes)
\Flash Disk\FreeFloat Access One\script.lua, OK (2624 bytes)
\Flash Disk\FreeFloat Access One\extra.lua, OK (1532 bytes)
Creating shortcut \Flash Disk\FreeFloat Access One\Freefloat Access.lnk
Shortcut created successfully.
Creating shortcut \Windows\Programs\Freefloat Access.lnk
Shortcut created successfully.
Installation successful.
Install time: 9 seconds

Above you see a typical installation.
CEInstaller connects to the CE device through ActiveSync.

If you already have installed the emulator and just wants to download a new configuration, select *Configuration* in the Files listbox.

During the installation process the application folder is created and files are transferred.

If the emulator has been previously installed and is running on the device, you get the error message *Failed to create file ... on CE device. Maybe the emulator is running?*.
Exit the emulator on the device and retry the installation. If the problem persists, reboot the CE device, preferably with a hardware reset button.

The emulator is installed in Flash memory (on selected devices). Even if the unit runs out of batteries or is cold-booted you don't need to re-install the emulator. You might have to re-configure the RF card though.

What is transferred during installation and where is it?

The CEInstaller installs the files from the directory on your PC that corresponds to the machine selected in the Configuration Application.

For example, the files installed to a Denso BHT-200 are located in

C:\Freefloat\Freefloat Access xxx\Denso\BHT200

xxx above reflect the version number of Freefloat Access.

As an example, let's add a script to the folder for the BHT-200.

1. Start Notepad.
2. Enter the following text:

```
function myTestFunction()
  app.send( "Hello there!" )
end
```

3. Save it as Test.lua in the folder
   C:\Freefloat\Freefloat Access xxx\Denso\BHT200.
   (The path might be different depending on where you installed the product on your PC.)
4. Start CEInstaller and select *Scripts* in the listbox Files. Then click on the Install button.
5. As you can see the script file you just created is transferred to the CE device.

Keyboard Mapping

Key mappings are defined in the script file **keytables.lua**. It contains a table for VT/ANSI keys (**keyTableAnsi**) and a table for 5250 and 3270 emulation (**keyTable5250**).

If line mode buffering is used in VT/ANSI, please note that all characters generated from a key mapping are not sent directly to host. Append {flush} to send the characters in the key mapping directly to the host. For example:

```
["KEYF5"]= "\027OT{flush}",
```

How to map a key

1. Open the script file **keytables.lua** in a text editor from the folder of the device you a working with.
2. Let's say you are using 5250 emulation and want to configure F1 to send the text "HELLO" and then press the Enter key.
3. In the text editor, find the definition for KEYF1 in the table keyTable5250 and change it to this:

   ["KEYF1"] = "HELLO{enter}",

   KEYF1 is the key identifier that identifies the hardware key. Inside a key mapping keys are denoted by enclosing its name in curly brackets {}. For all key identifiers and key names, please see **Key Table** below.

   In 3270 and 5250 the key name ({enter}) means the action performed by the emulation when that key is pressed. That is, if KEYENTER is mapped to generate the text "hello" then {enter} in another key mapping still performs the actual Enter action.

Note that a key mapping can contain a sequence of text and/or multiple keys.

Key Table

Key Identifier are the internal names of keys or key combinations. This name is passed to the script event function **onKey()**. When you need to refer to a key in a custom key mapping, you should use Key Name or Key Name Alias.

Key Identifier	Key Name	Key Name Alias
KEYCTRLA	{ctrl-a}	
KEYCTRLB	{ctrl-b}	
KEYCTRLC	{ctrl-c}	
KEYCTRLD	{ctrl-d}	
KEYCTRLE	{ctrl-e}	
KEYCTRLF	{ctrl-f}	
KEYCTRLG	{ctrl-g}	
KEYCTRLH	{ctrl-h}	
KEYCTRLI	{ctrl-i}	
KEYCTRLJ	{ctrl-j}	
KEYCTRLK	{ctrl-k}	
KEYCTRLL	{ctrl-l}	
KEYCTRLM	{ctrl-m}	
KEYCTRLN	{ctrl-n}	
KEYCTRLO	{ctrl-o}	
KEYCTRLP	{ctrl-p}	
KEYCTRLQ	{ctrl-q}	
KEYCTRLR	{ctrl-r}	
KEYCTRLS	{ctrl-s}	
KEYCTRLT	{ctrl-t}	
KEYCTRLU	{ctrl-u}	
KEYCTRLV	{ctrl-v}	
KEYCTRLW	{ctrl-w}	
KEYCTRLX	{ctrl-x}	
KEYCTRLY	{ctrl-y}	
KEYCTRLZ	{ctrl-z}	
KEYTAB	{tab}	
KEYSHIFTTAB	{shift-tab}	
KEYESC	{escape}	{esc}
KEYENTER	{enter}	
KEYBACKSPACE	{backspace}	
KEYF1	{pf1}	{f1}
KEYF2	{pf2}	{f2}
KEYF3	{pf3}	{f3}

KEYF4	{pf4}	{f4}
KEYF5	{pf5}	{f5}
KEYF6	{pf6}	{f6}
KEYF7	{pf7}	{f7}
KEYF8	{pf8}	{f8}
KEYF9	{pf9}	{f9}
KEYF10	{pf10}	{f10}
KEYF11	{pf11}	{f11}
KEYF12	{pf12}	{f12}
KEYF13	{pf13}	{f13}
KEYF14	{pf14}	{f14}
KEYF15	{pf15}	{f15}
KEYF16	{pf16}	{f16}
KEYF17	{pf17}	{f17}
KEYF18	{pf18}	{f18}
KEYF19	{pf19}	{f19}
KEYF20	{pf20}	{f20}
KEYF21	{pf21}	{f21}
KEYF22	{pf22}	{f22}
KEYF23	{pf23}	{f23}
KEYF24	{pf24}	{f24}
KEYSHIFTF1	{shift-f1}	
KEYSHIFTF2	{shift-f2}	
KEYSHIFTF3	{shift-f3}	
KEYSHIFTF4	{shift-f4}	
KEYSHIFTF5	{shift-f5}	
KEYSHIFTF6	{shift-f6}	
KEYSHIFTF7	{shift-f7}	
KEYSHIFTF8	{shift-f8}	
KEYSHIFTF9	{shift-f9}	
KEYSHIFTF10	{shift-f10}	
KEYSHIFTF11	{shift-f11}	
KEYSHIFTF12	{shift-f12}	
KEYCTRLF1	{ctrl-f1}	
KEYCTRLF2	{ctrl-f2}	
KEYCTRLF3	{ctrl-f3}	
KEYCTRLF4	{ctrl-f4}	
KEYCTRLF5	{ctrl-f5}	
KEYCTRLF6	{ctrl-f6}	
KEYCTRLF7	{ctrl-f7}	

KEYCTRLF8	{ctrl-f8}
KEYCTRLF9	{ctrl-f9}
KEYCTRLF10	{ctrl-f10}
KEYCTRLF11	{ctrl-f11}
KEYCTRLF12	{ctrl-f12}
KEYALTF4	{alt-f4}
KEYLEFT	{left}
KEYRIGHT	{right}
KEYUP	{up}
KEYDOWN	{down}
KEYSHIFTLEFT	{shift-left}
KEYSHIFTRIGHT	{shift-right}
KEYSHIFTUP	{shift-up}
KEYSHIFTDOWN	{shift-down}
KEYALTLEFT	{alt-left}
KEYALTRIGHT	{alt-right}
KEYALTUP	{alt-up}
KEYALTDOWN	{alt-down}
KEYNUMDIVIDE	{numdivide}
KEYNUMMULTIPLY	{nummultiply}
KEYINS	{insert}
KEYDEL	{delete}
KEYEND	{end}
KEYHOME	{home}
KEYPGDN	{pagedown}
KEYPGUP	{pageup}
KEYNUMSUBTRACT	{numsubtract}
KEYNUMADD	{numadd}
KEYCTRLALTC	{ctrl-alt-c}
KEYNUMENTER	{numenter}
KEYALTDEL	{alt-delete}
KEYALTP	{alt-p}
KEYCTRLPGUP	{ctrl-pageup}
KEYALTNUMADD	{alt-numadd}
KEYALTNUMSUBTRACT	{alt-numsubtract}
KEYCTRLNUMADD	{ctrl-numadd}
KEYCTRLNUMSUBTRACT	{ctrl-numsubtract}
KEYALTHOME	{alt-home}
KEYCTRLHOME	{ctrl-home}
KEYCTRLALTSHIFTC	{ctrl-alt-shift-c}

KEYSHIFTINSERT	{shift-insert}
KEYCTRLDELETE	{ctrl-delete}
KEYCTRLALTP	{ctrl-alt-p}
KEYCTRLALTH	{ctrl-alt-h}
KEYALT1	{alt-1}
KEYALT2	{alt-2}
KEYALT3	{alt-3}
KEYPAUSE	{pause}
KEYCTRLBREAK	{ctrl-break}
KEYSHIFTESC	{shift-esc}
KEYLSHIFT	{left-shift}
KEYRSHIFT	{right-shift}
KEYLCTRL	{left-ctrl}
KEYRCTRL	{right-ctrl}
KEYLALT	{left-alt}
KEYRALT	{right-alt}
KEYALTGR	{altgr}
KEYAPPS	{apps}
KEYM1[1]	{m1}
KEYM2[1]	{m2}
KEYM3[1]	{m3}
KEYSHIFTM1[1]	{shift-m1}
KEYSHIFTM2[1]	{shift-m2}
KEYSHIFTM3[1]	{shift-m3}
KEYSEND[2]	{send}

[1] Unique keys on Denso BHT handhelds
[2] Unique key on Denso BHT-400

Local Action Keys

There are several keys mapped to local actions in Freefloat Access, for example ALT + arrow keys which scrolls the viewport. The actions for these keys are performed in the script event method **onKey()**. The table below lists the keys and the action performed by the key. These actions can be modified or removed by changing the **onKey()** method.

Key Identifier	Action
KEYALTF4	Exit Freefloat Access
KEYALTNUMADD	Increases the font height
KEYALTNUMSUBTRACT	Decreases the font height
KEYCTRLNUMADD	Increases the font width
KEYCTRLNUMSUBTRACT	Decreases the font width
KEYALTLEFT	Scrolls the viewport left
KEYALTRIGHT	Scrolls the viewport right
KEYALTUP	Scrolls the viewport up
KEYALTDOWN	Scrolls the viewport down
KEYALTP	Toggles the on-screen keyboard
KEYM1	Toggles the on-screen keyboard (Denso BHT)

Emulation Actions

These keys correspond to actions in the emulations. There are physical key combinations that are mapped to these actions. Please refer to **keytables.lua** to find out the mappings.

Key Name	Action	Emulation
{backtab}	Move cursor to the previous field	3270/5250
{newline}	Move cursor to the first field on the next row	3270/5250
{fieldminus}	Set the field sign character to a minus	5250
{fieldplus}	Set the field sign character to a plus	5250
{fieldexit}	Move cursor to the next field	5250
{clearfield}	Clear the contents of the current field	3270/5250
{deleteword}	Deletes the word at the cursor	3270
{attention}	Performs the Attention action	3270/5250
{sysreq}	Performs the SysReq action	3270/5250
{reset}	Performs the Reset action (unlocks the keyboard if locked)	3270/5250
{clear}	Performs the Clear action	3270/5250
{dup}	3270: Insert Mode, copies and insert the character under the cursor 5250: Fills the remaining space in the current field with DUP characters and moves the cursor to the next field	3270/5250
{erase}	Erases data in all fields and move the cursor to the first field	3270
{print}	Performs the Print action	5250
{help}	Performs the Help action	5250
{pa1}	Performs the PA1 action	3270/5250
{pa2}	Performs the PA2 action	3270/5250
{pa3}	Performs the PA3 action	3270/5250
{flush}	Sends the contents of the line buffer when in line mode	VT/ANSI

Configurable Keyboard

The on-screen keyboard is configurable. To modify the default keyboard layouts or to create a new keyboard layout, please download Freefloat Key and use the Designer application. You do not need a license for Freefloat Key to be able to use the Designer.

Please refer to the Freefloat Key documentation on how to use the Designer.

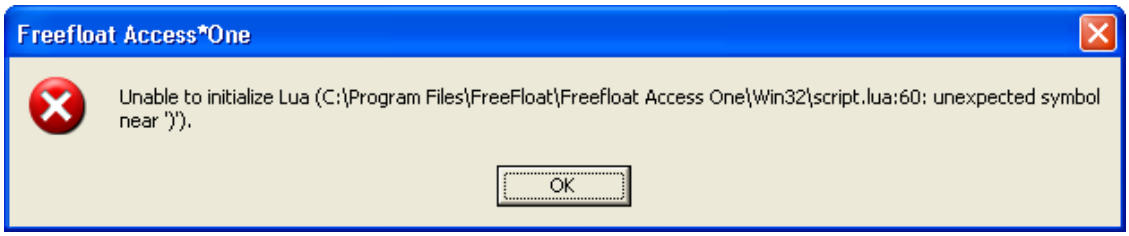
Scripting

Overview

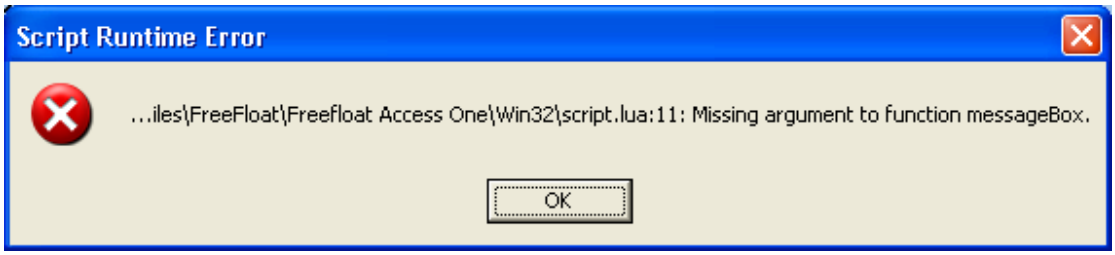
Scripting is a means to automate things, such as logging on to a host, or navigate a menu system to the desired user application etc. It can also be used to change the behavior of the emulator.

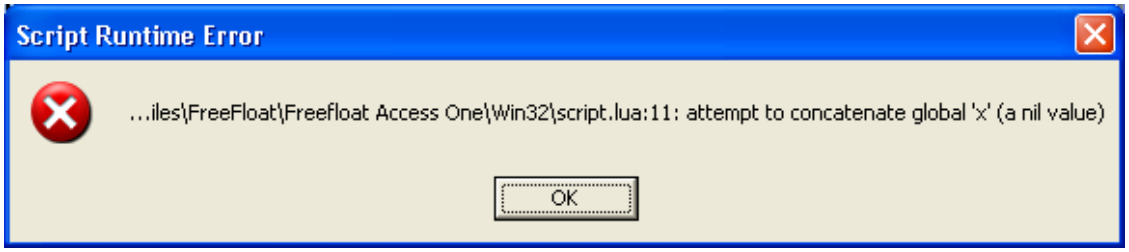
The name of the script file is Script.lua. A copy of the default script is placed in the platform folder for each device. To modify the script, edit the correct Script.lua with a text editor, for example Notepad. Transfer the new script to your platform, for example a handheld device, and restart Freefloat Access to try out your new script.

If you make a mistake, for example create a syntax error, an error message is displayed when the script is compiled:


Freefloat Access*One
Unable to initialize Lua (C:\Program Files\FreeFloat\Freefloat Access One\Win32\script.lua:60: unexpected symbol near ')').
OK

Also, some errors can appear when the script is running, so called run-time errors. Here are a couple of examples:


Script Runtime Error
...iles\FreeFloat\Freefloat Access One\Win32\script.lua:11: Missing argument to function messageBox.
OK


Script Runtime Error
...iles\FreeFloat\Freefloat Access One\Win32\script.lua:11: attempt to concatenate global 'x' (a nil value)
OK

The Lua Language

From http://www.lua.org/about.html:

Lua is a powerful, fast, light-weight, embeddable scripting language.

Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics.

Put simply, Lua makes barcode processing and custom scripting in Freefloat Access very flexible and powerful.

The reference manual for Lua can be found at the Lua site:

http://www.lua.org/

There is also a printed book on the Lua language, called **Programming in Lua**, which is more accessible than the reference manual.

Apart from Lua and its built-in language, Freefloat Access exposes a number of useful methods to the script.

Script Events

When things happen in Freefloat Access, for example a key is pressed or a barcode is read, an event is generated. This results in a script method being called. The methods called when events happen are called *event methods*.

The table below is an overview and short description of all the different event methods. For a more detailed explanation, see the topic **Event Methods** below.

Event Handler	When Called
onKey	A key is pressed
onBarcode	A barcode is read (only possible on platforms with support for integrated barcode reader)
onStart	A connection to the host is established
onEnd	The connection to the host is closed
onActivate	The application window is activated
onDeactivate	The application window is deactivated
onCursorMove	The cursor is moved to a new position
onEvent	A timer event is fired
onNetworkData	Network data is received
onMouseClick	The mouse is clicked
onMouseWheel	The mouse scroll wheel is used

Event Methods

In this topic all the event methods are explained in detail.

onKey(keyname)

This method is called when a key is pressed.

This method receives a string argument containing the keyname or the character pressed.

Example: Beep when a certain key is pressed

```
function onKey( keyname )
  if keyname == "b" then
    -- Issue a short beep (3000 Hz, 50 ms)
    app.beep( 3000, 50 )
  end

  ...

end
```

onStart()

This method is called when a connection to the host is established.

This method receives no arguments.

Example: Beep on start

```
function onStart()
  -- Issue a short beep (3000 Hz, 50 ms)
  app.beep( 3000, 50 )
end
```

onEnd()

This method is called when a connection to the host is ended. The connection can be ended for a number of different reasons:

- The host closes the connection upon logging out from the system
- The user selects the disconnect command
- The connection is closed by a script command
- A network error breaks the connection

Example: Beep on end

```
function onEnd()
  -- Issue a short beep (1000 Hz, 50 ms)
  app.beep( 1000, 50 )
end
```

onActivate()

This method is called when the Freefloat Access application window is activated.

This method receives no arguments.

Example: Play a sound when activated

```
function onActivate()
  app.playSound( "click.wav" )
end
```

onDeactivate()

This method is called when the Freefloat Access application window is deactivated.

This method receives no arguments.

Example: Play a sound when deactivated

```
function onDeactivate()
  app.playSound( "click.wav" )
end
```

onCursorMove()

This method is called when cursor is moved to a new position. The movement can be initiated from the host or locally by the user.

This method receives no arguments.

Example: Send a text string triggered by a certain cursor position

```
function onCursorMove()
  row, col = app.getCursorRowCol()
  if row == 10 and col == 2 then
    app.send( "at row 10 and column 2" )
  end
end
```

onEvent(eventname)

This method is called when a timer event is triggered. Timer events are started by calling the script method **app.startEvent()**.

This method receives a string argument containing the timer event name.

Example: Send the event name

```
function onEvent( eventname )
  app.send( eventname )
end
```

onNetworkData()

This method is called when data is received from the host.

This method receives no arguments.

Example: Issue a short beep when data is received

```
function onNetworkData()
  app.beep( 1000, 1 )
end
```

onMouseClick()

This method is called when the user left-clicks the mouse in the Freefloat Access text area. By default, this event method implements the hot spot function.

This method receives no arguments. This method should return true if it processes the mouse click and false otherwise.

Example: The default implementation

```
-- Return true if handled, return false for default behaviour (viewport scrolling)
function onMouseClick( row, col )
  local strkey = keyTableHotSpot[ string.upper( app.getHotspotText( row, col ) ) ]
  if strkey ~= nil then
    app.send( strkey )
    return true
  end

  return false
end
```

onMouseWheel(delta, shift, control)

This method is called when the user scrolls the mouse wheel. By default, this event method implements the font resize function.

This method receives three arguments. The argument delta contains the relative movement of the scroll wheel. A positive value in delta means forward scroll and a negative a backward scroll. The arguments shift and control indicates the status of whether the corresponding keys are held down.

Example: The default implementation

```
function onMouseWheel( delta, shift, control )
  delta = delta / 120
  if control then
    if delta < 0 then
      app.decreaseFontWidth( -delta )
    else
      app.increaseFontWidth( delta )
    end
  else
    if delta < 0 then
      app.decreaseFontHeight( -delta )
    else
      app.increaseFontHeight( delta )
    end
  end
end
```

Script Methods

Lua is a generic script language and has methods to manipulate string, tables, files, and so on. In order to make Lua useful in Freefloat Access, a number of functions has been exposed to the script engine. These functions allow a script to retrieve text from the emulation text buffer, send text and keys to the host, control the on-screen keyboard, and so on.

The tables below offer an overview of the methods. The methods have been grouped into areas of interest for easier reference. Below the tables there is a reference section with a detailed description of each method.

When these methods are used in a script, you need to prefix them with "app.", for example **app.sleep(100)**.

Output	
Method	**Description**
beep	Beeps with the internal speaker
log	Adds a message to the log file
messageBox	Displays a message box
playSound	Plays a sound file
send	Sends keyboard data

Text Buffer	
Method	**Description**
getBufferText	Retrieves text from the emulation screen buffer
getHotspotText	Retrieves a word from the specified text position
setBufferText	Puts text into the emulation screen buffer

Cursor	
Method	**Description**
getCursorRowCol	Gets the cursor position
setCursorRowCol	Sets the cursor position

Scanner and Barcode	
Method	**Description**
gs1Decode	Parses the contents of a GS1-128/UCC/EAN-128 code
enableScanner	Enables the barcode scanner
disableScanner	Disables the barcode scanner

Viewport	
Method	**Description**
findCursor	Scrolls the viewport making the cursor visible
lockWindow	Disables manual scrolling of the text viewport
unlockWindow	Enables manual scrolling of the text viewport
isWindowLocked	Determines whether manual scrolling is enabled or not
setOrigin	Sets the text viewport offset
getOrigin	Gets the text viewport offset
viewportLeft	Scrolls the text viewport one character to the left, right, up, and down
viewportRight	
viewportUp	
viewportDown	

Emulation Type	
Method	**Description**
getEmulation	Gets the active emulation type

Time & Timer Events	
Method	**Description**
sleep	Suspends script execution for the specified time
getTickCount	Gets the system timer tick
getTime	Get the system date and time
setScriptTimeout	Sets the script timeout
getScriptTimeout	Gets the script timeout
startEvent	Starts a timer event
stopEvent	Stops a timer event

Session Control	
Method	**Description**
terminate	Terminates the host session
restart	Terminates the host session and start a new session
reboot	Reboots the PC

Application Launch	
Method	**Description**
run	Starts a program
isAppRunning	Determines if an application is still running
closeAppHandle	Closes the application handle

Font	
Method	**Description**
getFontSize	Gets the current font size
setFontSize	Sets a new font size
setFontWeight	Sets a new font weight
increaseFontHeight	Increases and decreases the font width and height
decreaseFontHeight	
increaseFontWidth	
decreaseFontWidth	

On-screen Keyboard Control	
Method	**Description**
showKeyboard	Shows the on-screen keyboard
hideKeyboard	Hides the on-screen keyboard
isKeyboardVisible	Determines if the on-screen keyboard is visible
showCanvas	Shows a canvas
hideCanvas	Hides a canvas
showAllCanvases	Shows all canvases
hideAllCanvases	Hides all canvases
setCanvasPosition	Moves a canvas
snapCanvas	Moves a canvas to a pre-determined position
getNumberOfCanvases	Gets the number of canvases in the keyboard layout
getCanvasXPos	Gets the horizontal position of a canvas
getCanvasYPos	Gets the vertical position of a canvas
getCanvasWidth	Gets the width of a canvas
getCanvasHeight	Gets the height of a canvas
isCanvasVisible	Determines if a canvas is visible

Freefloat Link Integration	
Method	**Description**
link_subscribe	Subscribes to data events from Freefloat Link
link_unsubscribe	Unsubscribes to data events from Freefloat Link
link_send	Sends data to Freefloat Link

Output

This section contains methods that generate some kind of output, for example text, keys, and sound.

beep(frequency, duration)

Description

Makes the speaker beep with the specified *frequency* and *duration*. Note that not all platforms supports beeping.

Arguments

Argument	Type	Description
frequency	Number	The frequency of the beep in Hertz.
duration	Number	The duration of the beep in milliseconds.

Returns

Nothing.

Example

```
-- Issue a short beep (3000 Hz, 50 ms)
function onStart()
  app.beep( 3000, 50 )
end
```

log(message)

Description

Adds a message to the application log file. This method could be useful for troubleshooting and testing scripts.

Arguments

Argument	Type	Description
message	String	The message to write to the log.

Returns

Nothing.

Example

```
-- Log keys and characters
function onKey( keyname )
  app.log( "Key pressed: " .. keyname )
  .
  .
  .
end
```

messageBox(title, message, type)

Description

Displays a message box with the specified *title* and *message*. The *type* argument specifies the number and type of buttons used.

Arguments

Argument	Type	Description
title	String	The message box title.
message	String	The message to be displayed.
type	Number, optional	The type of buttons to use.

Returns

A value from the constants in mbResults (see below) indicating the button that was clicked.

Constants

Use one or more of the following constants for the *type* argument:

```
mbTypes =
{
  ["OK"] = 0,
  ["OKCANCEL"] = 1,
  ["YESNOCANCEL"] = 3,
  ["YESNO"] = 4,
  ["ICONHAND"] = 16,
  ["ICONQUESTION"] = 32,
  ["ICONEXCLAMATION"] = 48,
  ["ICONASTERISK"] = 64,
  ["DEFBUTTON1"] = 0,
  ["DEFBUTTON2"] = 256,
  ["DEFBUTTON3"] = 512,
}
```

The possible return values are:

```
mbResults =
{
  ["OK"] = 1,
  ["CANCEL"] = 2,
  ["YES"] = 6,
  ["NO"] = 7,
}
```

Example

```
-- Ask the user if she/he wants to exit Freefloat Access and acts on the answer
function onKey( name )
  if keyname == "KEYF1" then
    local result = app.messageBox( "Script", "Are you sure you want to exit?",
      mbTypes["YESNO"] + mbTypes["DEFBUTTON2"] )

    if mbResults["YES"] == result then
      app.terminate( false )
    end
  end

  ...
end
```

playSound(filename, options)

Description

Plays the sound specified by the *filename* argument. Sound files should be in .wav format and be placed in the application folder. If you pass "normal", "barcode", or "error" as the filename argument, the internal beeper will be used and play the associated sounds defined by the configuration options below:

Common\Misc\Bell
Common\Barcode\Read Confirmation
Common\Misc\Error Tone

Please note that this only applies to devices/platforms that has a beeper device.

Arguments

Argument	Type	Description
filename	String	The path and filename of the sound file.
options	Number	Specifies the behavior for the sound playback. Use the constants in *soundOptions*.

Returns

Nothing.

Constants

```
soundOptions =
{
  ["SYNC"] = 0,
  ["ASYNC"] = 1,
  ["NODEFAULT"] = 2,
  ["LOOP"] = 8,
  ["NOSTOP"] = 16,
  ["NOWAIT"] = 8192,
}
```

Example

```
function onBarcode( data )
  app.playSound( "mybarcodesound.wav" )
  app.send( trimControlChars( data ) .. "{enter}", true )
end
```

send(data, fast)

Description

Sends keyboard data.

Arguments

Argument	Type	Description
data	String	The data to send.
fast	Boolean	Indicates if the data should be processed fast, i.e. screen update is turned off while sending.

The argument *data* is a string consisting of text and/or key names.

Type	Sample String	Description
Text	"Rob was here"	Regular characters.
Key Name	"{left}"	Key names corresponds to the keys defined in the sections **Key Table**, **Local Action Keys**, and **Emulation Actions**.
Character	"\065"	Character in decimal notation.

Returns

Nothing.

Example

```
function onMouseClick( row, col )
  -- Send a regular string
  app.send( "Rob was here!" )

  -- Send the character A
   app.send( "\065" )

  -- Send enter using its key name
  app.send( "{enter}" )

  return true
end
```

Note!

- When you want **send()** to send the character \ you need to escape that with an extra backslash.
- An error message will be displayed if you specify a key name for a key that does not exist.

Text Buffer

This section contains methods that accesses the emulation text buffer. These methods are useful when writing scripts that should act upon what is displayed on the screen.

getBufferText(row, col, length)

Description

Returns the text at the specified buffer position. This method is useful when you need to detect which screen is currently being displayed.

Arguments

Argument	Type	Description
row	Number	Specifies the position in the text buffer
col	Number	
length	Number	The number of characters to return

Returns

The text string.

Example

```
function onMouseClick( row, col )
  -- Display ten characters at the clicked buffer position
  app.messageBox( "Text in buffer", app.getBufferText( row, col, 10 ) )
  return true
end
```

setBufferText(row, col, text)

Description

Sets the text at the specified buffer position.

Arguments

Argument	Type	Description
row	Number	Specifies the position in the text buffer
col	Number	
text	String	The text to put into the screen buffer

Returns

nothing

Example

```
function onMouseClick( row, col )
  -- Enter the string "New Text" at the clicked buffer position
  app.setBufferText(row,col,"New Text")
  return true
end
```

getHotspotText(row, col)

Description

Returns the word at the specified buffer position.

This method is used internally in the function onMouseClick(row, col) to implement clickable function keys, also called hotspots.

The method scans left and right from the selected position to build the returned word.
All letters (including international letters) and digits are included in the returned word.

Other characters, like hyphen or slash, will terminate the hotspot word scan.
For example, if you click on the letter y in the word `IBM-system` the returned hotspot word will be system and not IBM-system.

Arguments

Argument	Type	Description
row	Number	Specifies the position in the text buffer
col	Number	

Returns

A string containing the word.

Example

```
function onMouseClick( row, col )
  -- Display the string at the clicked buffer position
  app.messageBox( "Text in buffer", app.getHotspotText( row, col ) )
  return true
end
```

Cursor

This section contains methods that gets or sets the cursor position.

getCursorRowCol( )

Description

Returns the current cursor position.

Arguments

None.

Returns

The cursor's position, row and column. Please note that there are two return values.

Example

```
function onMouseClick( row, col )
  local r, c = app.getCursorRowCol()
  app.messageBox( "Cursor Position", "Row: " .. r .. "\n" .. "Column: " .. c )
  return true
end
```

setCursorRowCol(row, col)

Description

Moves the cursor to the specified position. Please note that this method only works in 3270 and 5250 emulation.

Arguments

Argument	Type	Description
row	Number	Specifies the position of the cursor.
col	Number	

Returns

Nothing.

Example

```
function onMouseClick( row, col )
  -- Move the cursor to the start of the tenth row
  app.setCursorRowCol( 10, 1 )
  return true
end
```

Scanner and Barcode

This section contains methods related to barcodes.

gs1Decode(data, strict)

Description

Parses the contents of a GS1-128 code (earlier called UCC-128 or EAN-128). For variable length fields followed by another field, the data must be delimited by a Group Separator (GS, ASCII 29, hex 1D).

Please refer to http://www.gs1.org for information about GS1 Application Identifiers.

Arguments

Argument	Type	Description
data	String	The GS1-128 data to be parsed and split into separate fields.
strict	Boolean	In strict mode, spaces are not allowed in alphanumeric fields.

Returns

A table where the keys are the Application Identifiers (AIs) and the values are the contents of the fields. If the parsing fails, a nil value is returned. The parsing can fail if the code is not a GS1 code or if the code doesn't follow the standard.

Constants

None.

Example

```
-- Parses a code and outputs a list of AIs and values
function onBarcode( data )
  -- Parse the GS1 code
  fields = app.gs1Decode( data, true )

  if fields then
    -- Output AIs and values
    for k,v in pairs( fields ) do
      app.send( "AI: " .. k .. "  Value: " .. v .. "{tab}" )
    end
  else
    app.messageBox( "Freefloat Access Script", "GS1 parsing failed." )
  end
end
```

enableScanner()

Description

Enables the barcode scanner. Please note that this method only has effect on devices with an integrated barcode scanner or if Freefloat Link is in subscription mode.

Arguments

None.

Returns

Nothing.

Example

```
-- Disable scanning when the cursor is on the tenth row
function onCursorMove()
  row, col = app.getCursorRowCol()
  if row == 10 then
    app.disableScanner()
  else
    app.enableScanner()
  end
end
```

disableScanner()

Description

Disables the barcode scanner. Please note that this method only has effect on devices with an integrated barcode scanner.

Arguments

None.

Returns

Nothing.

Example

See **enableScanner()** above.

Viewport

This section contains methods that deal with the viewport.

findCursor()

Description

Scrolls the viewport making the cursor visible. This method has no effect if the viewport is locked.

Arguments

None.

Returns

Nothing.

lockWindow()

Description

Disables manual scrolling of the text viewport. A script can change the origin of the viewport by using **app.setOrigin()**.

Arguments

None.

Returns

Nothing.

unlockWindow()

Description
Enables manual scrolling of the text viewport.

Arguments
None.

Returns
Nothing.

isWindowLocked()

Description
Determines whether manual scrolling is enabled or not.

Arguments
None.

Returns
true if manual scrolling is turned off, **false** otherwise.

setOrigin(row, col)

Description

Sets the text viewport offset.

Arguments

Argument	Type	Description
row	Number	Specifies the new offset of the text viewport
col	Number	

Returns

Nothing.

Example

```
function onKey( keyname )
  -- Set the viewport offset to be the upper left corner
  if keyname == "KEYF1" then
    app.setOrigin( 1, 1 )
    return
  end

  ...
end
```

getOrigin()

Description

Gets the text viewport offset.

Arguments

None.

Returns

The text viewport's offset, row and column. Please note that there are two return values.

Example

```
function onMouseClick( row, col )
  local r, c = app.getOrigin()
  app.messageBox( "Viewport Offset", "Row: " .. r .. "\n" .. "Column: " .. c )
  return true
end
```

viewportLeft(), viewportRight(), viewportUp(), viewportDown()

Description

Scrolls the text viewport one character to the left, right, up, and down. These methods have no effect if manual scrolling of the viewport has been disabled.

Please see the function **onKey()** in the standard script for an example of how these methods are used.

Arguments

None.

Returns

Nothing.

Emulation Type

getEmulation()

Description

Gets the active emulation type.

Arguments

None.

Returns

A string indicating the active emulation type. Can be one of the following: "ANSI", "3270", "5250". For 3270 and 5250 emulation, "ANSI" may be returned if **app.getEmulation()** is called early during the connection phase.

Example

```
function onStart()
  app.messageBox( "Emulation Type", app.getEmulation() )
end
```

sleep(delay)

Description

Suspends script execution for the specified time. While sleeping, data to and from the server is processed.

Arguments

Argument	Type	Description
delay	Number	Specifies the delay in milliseconds

Returns

Nothing.

Example

```
function onKey( keyname )
  if keyname == "KEYF1" then
    app.send( "username{tab}" )
    app.sleep( 1000 )
    app.send( "password{enter}" )
    return
  end
  .
  .
  .
end
```

getTickCount()

Description

Gets the system timer tick.

Arguments

None.

Returns

The system timer tick.

Example

```
function onStart()
  app.messageBox( "Timer Tick is: ", app.getTickCount() )
end
```

getTime()

Description

Get the system date and time.

Arguments

None.

Returns

A table containing the current date and time.

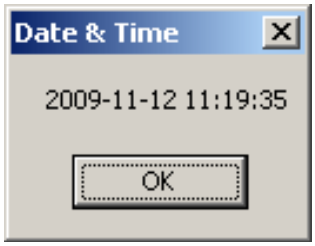
Field name	Content
["year"]	year,with four digits
["month"]	month, January = 1, February = 2, and so on
["day"]	day of the month, 1-31
["dayofweek"]	day of the week. Sunday = 0, Monday = 1, and so on
["hour"]	hour
["minute"]	minute
["second"]	second
["milliseconds"]	milliseconds

Example

```
function onKey( keyname )
  if keyname == "KEYCTRLF12" then
      local t = app.getTime()

      s = string.format( "%04d-%02d-%02d %02d:%02d:%02d",
      t["year"],t["month"],t["day"],t["hour"],t["minute"],t["second"] )
     app.messageBox( "Date & Time", s ,mbTypes["OK"] )
     return
  end
  .
  .
end
```

Pressing CTRL-F12 displays the current date & time like this:



setScriptTimeout(timeout)

Description

The timeout is used to detect and stop a “runaway” script, i.e. a script function that never finishes.

The default timeout is 5 seconds, set in the onStart() function in the script.

A value of 0 disables the timeout function (not recommended).

Arguments

Timeout value in milliseconds.

Returns

Nothing.

Example

```
function onStart()
  app.setScriptTimeout( 5000 )
  .
  .
end
```

getScriptTimeout()

Description

Gets the current script timeout value.

One use might be when you need to change or disable the timeout for some activity that takes a long time to finish but need to restore the timeout to what it was after that operation is completed.

Arguments

None

Returns

The current timeout value.

Example

```
function someFunction()
  local scripttimeout = app.getScriptTimeout()

  -- Do some time consuming activity

  app.setScriptTimeout( scripttimeout )
end
```

startEvent(eventname, interval, repeat)

Description

Starts a timer event.
You don't need to call stopEvent() to change the interval of an already started timer.
When the timer expires the script function onEvent() is called.

Arguments

Argument	Type	Description
eventname	String	The name of the event.
interval	Numeric	The timer event interval in milliseconds. The timer will expire *interval* mS from the time you call this function.
repeat	Boolean	Specifies whether the event should be auto-repeating (true) or if it is a one-shot event (false).

Returns

Nothing.

Example

```
function onStart()
  app.startEvent( "A", 5000, true )
end
```

stopEvent(eventname)

Description

Stops a timer event.

Arguments

Argument	Type	Description
eventname	String	The name of the event to stop.

Returns

Nothing.

Example

```
function onEnd()
  app.stopEvent( ”A” )
end
```

Session Control

terminate(ask)

Description

Terminates a host session and exits.

Arguments

Argument	Type	Description
ask	Boolean	Specifies whether a question should be displayed or not.

Returns

Nothing.

Example

```
function onKey( keyname )
  if keyname == "KEYALTF4" then
    app.terminate( true )
  end
  .
  .
```

restart()

Description

Terminates the current host session and starts a new host session.

Arguments

None

Returns

Nothing.

Example

```
function onKey( keyname )
  if keyname == "KEYALTF5" then
    app.restart()
  end
  .
  .
```

reboot(forced)

Description

Reboots the PC.

Arguments

Argument	Type	Description
forced	Boolean	Specifies whether a forced reboot should be performed.

Returns

Nothing.

Application Launch

run(path, arguments, delay)

Description

Starts a program.

Arguments

Argument	Type	Description
path	String	Full path name to the program to run.
arguments	String (optional)	Command line arguments to pass to the program.
delay	Number (optional)	Time in milliseconds to wait until the started program is idle and awaiting user input.

Returns

The process handle of the started program. Used as an argument to the functions isAppRunning() and closeAppHandle().

If you don't need to keep track of the started application, you should call closeAppHandle() immediately to release the allocated handle.

Example

```
function onKey( keyname )
  if keyname == "KEYCTRLF11" then
    appHandle = app.run( "notepad", "c:\\testdoc.txt", 100 )
  end
  .
  .
```

isAppRunning(handle)

Description
Determines if an application is still running.

Argument
The process handle returned by the run() command.

Returns
True or false.

closeAppHandle(handle)

Description
Closes the process handle.

Argument
The process handle returned by the run() command.

Returns
Nothing.

Font

This section contains methods that deals with the font size.

getFontSize()

Description

Gets the current font size.

Arguments

None.

Returns

The height and width of the current font.

Example

```
function onMouseClick( row, col )
  -- Display the current font height and width
  local h,w = app.getFontSize()
  app.messageBox( "Font Size", "Height: " .. h .. " " .. "Width: " .. w )
  return true
end
```

setFontSize(height, width)

Description

Sets a new font size.

Arguments

Argument	Type	Description
height	Number	The height of the font
width	Number	The width of the font

Returns

Nothing.

setFontWeight(height)

Description

Sets a new font weight.

Arguments

Argument	Type	Description
weight	Number	The weight of the font

Sensible values for weight are 400 (normal), 600 (semi-bold), and 700 (bold).

Returns

Nothing.

increaseFontHeight(delta), decreaseFontHeight(delta)

increaseFontWidth(delta), decreaseFontWidth(delta)

Description

Increases/decreases the font width/height by the specified amount.

These functions are used by the event method **onMouseWheel()** in the default script.

Arguments

Argument	Type	Description
delta	Number	How much the width/height should be increased/decreased

Returns

Nothing.

On-screen Keyboard Control

This section contains methods that deals with the integrated Freefloat Key keyboard.

showKeyboard(), hideKeyboard()

Description

Shows or hides Freefloat Key.

Arguments

None.

Returns

Nothing.

isKeyboardVisible()

Description
Determines if the on-screen keyboard is visible.

Arguments
None.

Returns
true if the keyboard is visble, **false** otherwise.

showAllCanvases(), hideAllCanvases()

Description
Show/hide all canvases of Freefloat Key.

Arguments
None.

Returns
Nothing.

showCanvas(canvas), hideCanvas(canvas)

Description
Show/hide a specific canvas of Fereefloat Key.

Arguments
The canvas number.

Returns
Nothing.

setCanvasPosition(canvas, xpos, ypos)

Description

Moves the specified canvas of Freefloat Key to a new position. The *xpos* and *ypos* arguments are screen coordinates.

Arguments

Argument	Type	Description
canvas	Number	The canvas number
xpos	Number	Specifies the new offset of the text viewport
ypos	Number	

Returns

Nothing.

snapCanvas(canvas, position)

Description

Moves the specified canvas of Freefloat Key to a pre-determined screen position.

Arguments

Argument	Type	Description
canvas	Number	The canvas number.
position	Number	The position to snap to, see table below for values.

Snap Position	Canvas Moved To
0	Upper Left
1	Upper Middle
2	Upper Right
3	Lower Left
4	Lower Middle
5	Lower Right

Returns

Nothing.

getNumberOfCanvases()

Description
Gets the number of canvases in the keyboard layout.

Arguments
None.

Returns
The number of canvases.

getCanvasXPos(canvas), getCanvasYPos(canvas)

getCanvasWidth(canvas), getCanvasHeight(canvas)

Description

Gets the horizontal position, vertical position, width or height of the specified canvas.

Arguments

The canvas number.

Returns

Function	Returns
getCanvasXPos	The horizontal position of the canvas
getCanvasYPos	The vertical position of the canvas
getCanvasWidth	The width of the canvas
getCanvasHeight	The height of the canvas

isCanvasVisible(canvas)

Description
Determines if the specified canvas is visible.

Arguments
The canvas number.

Returns
true if the canvas is visible, **false** otherwise.

Freefloat Link Integration

This section contains methods that deals with data integration with Freefloat Link.

link_subscribe()

Description

Subscribes to data events from Freefloat Link.

In a scenario where Freefloat Access subscribes to data from Freefloat Link, the data should be sent from Freefloat Link with the method *sendSubscriberData()*. The data will then be received as barcode data in the script function *onBarcode()*.

If an error occurs when this method is called (for example Freefloat Linke is not running), a runtime script error will be displayed.

Arguments

None.

Returns

Nothing.

link_unsubscribe()

Description
Unsubscribes to data events from Freefloat Link.

If a script tries to unsubscribe when not subscribing a runtime script error will be displayed.

Arguments
None.

Returns
Nothing.

link_send(data)

Description
Sends data to Freefloat Link.

Note that this will trigger the event **onExternalData()** in Freefloat Link.

Arguments
data is the string data to send.

Returns
Nothing.

Hotspots

Hotspot keys are handled by the script. When the user clicks or taps on the screen, the event method **onMouseClick()** is called. In the default script, this method checks if the word that the user clicked upon is in the hotspot key list. If it is, the action for the key is executed.

The hotspot key list is stored in the script file keytables.lua and is named keyTableHotSpot. You can either remove or add keys to this list to make a custom hotspot key list.

If you want to turn off the hotspot key function, modify the method **onMouseClick()** so it looks like this:

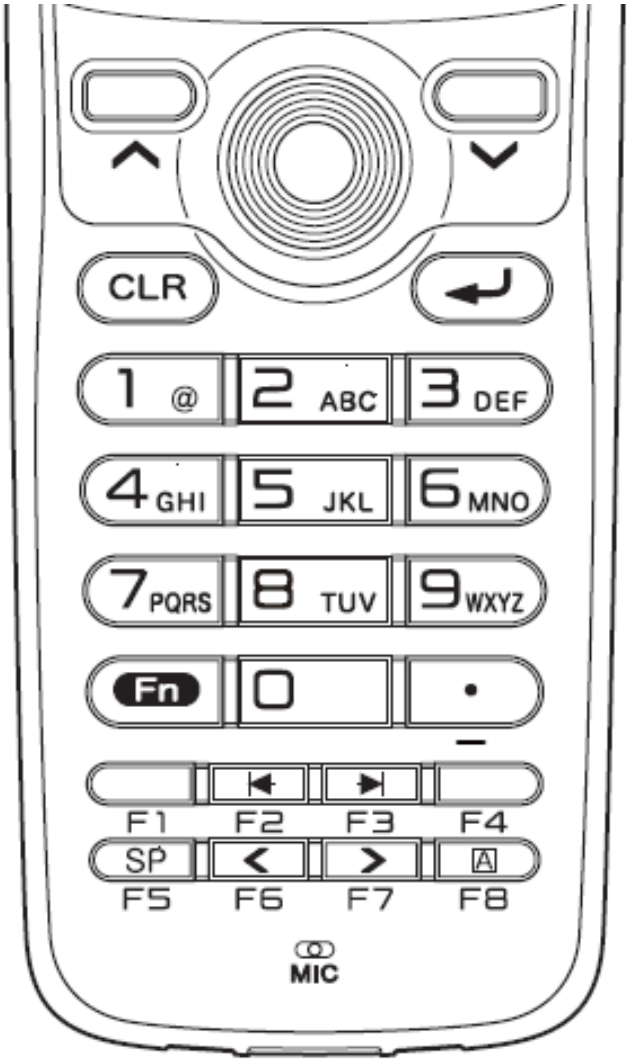
```
-- Return true if handled, return false for default behaviour (viewport scrolling)
function onMouseClick( row, col )
  return false
end
```

Appendix A – Model Specifics

Casio DT-X7

Keyboard

The Casio DT-X7 has no touch screen. To be able to navigate and use terminal specific keys, two extra keyboard modes has been added. Note that these modes are in addition to the standard modes of the keyboard (numeric, lower case, and uppercase). The standard modes are changed using the A key.



The two additional modes are called **Function key mode 1** (f1) and **Function key mode 2** (f2). The status bar will indicate which one of the five possible modes which is active.

You toggle between one of the standard modes and the extra keyboard modes by pressing the yellow key (F4). All keys but arrow keys are "single shot" keys. It means that the keyboard mode will go back to the standard mode after pressing Yellow and another key.

You can also cycle through the standard mode and the extra modes by pressing the Yellow key repeatedly.

Function Key Mode 1

Key Sequence	Action	Stays in mode
Yellow + CLR	Reset/unlock (5250/3270)	
Yellow + Enter	Centers the display around the cursor	
Yellow + 1 to 9	Sends F1 to F9 to the host	
Yellow + 0	Sends F10 to the host	
Yellow + up arrow	Scroll 1 line up	Yes
Yellow + down arrow	Scroll 1 line down	Yes
Yellow + left arrow	Scroll 1 column left	Yes
Yellow + right arrow	Scroll 1 column right	Yes

Function Key Mode 2

Key Sequence	Action	Stays in mode
Yellow + Yellow + Enter	Centers the display around the cursor	
Yellow + Yellow + 1 to 9	Sends F11 to F19 to the host	
Yellow + Yellow + 0	Sends F20 to the host	
Yellow + Yellow + up arrow	Increase font height	Yes
Yellow + Yellow + down arrow	Decrease font height	Yes
Yellow + Yellow + left arrow	Decrease font width	Yes
Yellow + Yellow + right arrow	Increase font height	Yes

Note! Red + CLR disconnects the session (a confirmation dialog will be displayed).

Casio DT-X8

Keyboard

The Left Multikey and the Right Multikey (marked L and R) have been assigned the keycodes "KEYAPP1" and "KEYAPP2".

Denso BHT-200

Keyboard

The M1 key toggles the keyboard. The M2 key works as the TAB key on a normal keyboard. Shift + M2 is the same as Shift + TAB.

On the BHT-200 with alphanumeric keyboard, you can activate the alpha mode by pressing and holding down the SF/ALP key until you hear a beep. The physical keyboard is then toggled between numeric and alpha input modes. The currently active mode is indicated by a N or an A in the status bar.

If you need quick access to the function keys F11 to F20 you can press and hold the SF (shift) key and then a function key. F1 is then F11, F2 is F12 and so on. Please note that this only works in numeric input mode.

You can save the RF setup by holding down the SF key and then powering off the unit. The RF setup will then survive even if the battery is completely drained.

Pasting

When running Freefloat Access on a BHT-200 together with kbifCE (supplied from Denso Wave) there is a config option that can speed up barcode handling: *Use Ctrl-V as paste command*. When enabled the key Ctrl-V is used as the paste command. When disabled Ctrl-V is used as a normal key, which in ANSI/VT emulation is treated as a control character.

Please note that you also need to choose the Send Mode *Use Clipboard* in kbifCE.

Using the paste method to transfer barcode data from kbifCE to Freefloat Access is approximately 45 times faster than using the kbifCE Send Mode *Keyboard Emulation*.

When kbifCE is not used, Freefloat Access has control over the barcode reader, and in this case the option has no effect.

Denso BHT-400

On a Denso BHT-400, Freefloat Access has the same indicator for numeric and alpha input modes as BHT-200. Please see more information above.

In addition, on BHT-400, an F is displayed when the keyboard function mode has been activated.

Symbol PPT8846

Please note that only the 15-key variant of the PPT8846 is supported.

The emulator remaps FUNC+keys to enable use of function keys. This remapping overrides any functions mapped in the operating system. The normal mappings are restored when you exit the emulator.

Machines that have a sound device capable of playing wav-files

In the folders for those devices there are some wave files. Feel free to replace them with your favorite sounds.

Filename	Event
startup.wav	When the emulator is started
normal.wav	When a new form with the sound alarm bit set is received in 3270/5250 emulation or when the bell characters is received in VT/ANSI emulation
barcode.wav	When a barcode is read
error.wav	When the keyboard is locked in 3270 or 5250 emulation

PSC Falcon 44xx with monochrome screen

The configuration for the PSC Falcon 44xx is intended for the color model. If you have a monochrome 44xx, insert these statements into the configuration file to get a black and white color scheme that is a bit more readable than the gray scale.

```
// Color Options
3270 Emulation\Colors\Default Background=16777215
5250 Emulation\Colors\Default Foreground=0
5250 Emulation\Colors\Default Background=16777215
VT/ANSI Emulation\Colors\Black=16777215
VT/ANSI Emulation\Colors\Blue=0
VT/ANSI Emulation\Colors\Green=0
VT/ANSI Emulation\Colors\Cyan=0
VT/ANSI Emulation\Colors\Red=0
```

```
VT/ANSI Emulation\Colors\Magenta=0
VT/ANSI Emulation\Colors\Brown=0
VT/ANSI Emulation\Colors\Light Gray=0
VT/ANSI Emulation\Colors\Dark Gray=0
VT/ANSI Emulation\Colors\Light Blue=0
VT/ANSI Emulation\Colors\Light Green=0
VT/ANSI Emulation\Colors\Light Cyan=0
VT/ANSI Emulation\Colors\Light Red=0
VT/ANSI Emulation\Colors\Light Magenta=0
VT/ANSI Emulation\Colors\Yellow=0
VT/ANSI Emulation\Colors\White=0

// Color Pair Options
3270 Emulation\Colors\Basic Colors\Unprotected Normal=0,16777215
3270 Emulation\Colors\Basic Colors\Protected Normal=0,16777215
3270 Emulation\Colors\Basic Colors\Unprotected Intensified=0,16777215
3270 Emulation\Colors\Basic Colors\Protected Intensified=0,16777215
5250 Emulation\Colors\Default Intensified=0,16777215
5250 Emulation\Colors\Green=0,16777215
5250 Emulation\Colors\White=0,16777215
5250 Emulation\Colors\Red=0,16777215
5250 Emulation\Colors\Turquoise=0,16777215
5250 Emulation\Colors\Yellow=0,16777215
5250 Emulation\Colors\Pink=0,16777215
5250 Emulation\Colors\Blue=0,16777215
3270 Emulation\Colors\Extended Colors\Blue=0,16777215
3270 Emulation\Colors\Extended Colors\Green=0,16777215
3270 Emulation\Colors\Extended Colors\Pink=0,16777215
3270 Emulation\Colors\Extended Colors\Red=0,16777215
3270 Emulation\Colors\Extended Colors\Turquoise=0,16777215
3270 Emulation\Colors\Extended Colors\White=0,16777215
3270 Emulation\Colors\Extended Colors\Yellow=0,16777215
3270 Emulation\Colors\Extended Colors\Default Normal=0,16777215
3270 Emulation\Colors\Extended Colors\Default Intensified=0,16777215
```

Appendix B – Logging Network Data

When the option *Raw Logging* is turned on, the emulator stores all network data into a file called `rawlog.txt`.

The log file can be used by an engineer for diagnostics.

Please note that raw logging can only be used when running Freefloat Access in regular Windows.

Raw Logging can be turned on and off via the Log menu. It can also be cleared and the formatted log file can be viewed from the menu.

Appendix C – EBCDIC Character Tables

The default code page used in the Freefloat Emulator is 278. Please see topic Display Character Translation on how to modify the translation tables.

Code Page 278, Swedish/Finnish

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	¢ SC040000	ä LA170000	å LA270000	É LE120000	0 ND100000
-1	(RSP) SP300000	` SD130000	/ SP120000	\ SM070000	a LA010000	j LJ010000	ü LU170000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC050000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	{ SM110000	ë LE170000	# SM010000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	[SM060000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	} SM140000	ï LI170000	$ SC030000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	é LE110000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	§ SM240000	¤ SC010000	ö LO170000	: SP130000	« SP170000	ª SM210000	¡ SP030000	¬ SM660000	(SHY) SP320000	1 ND011000	2 ND021000	3 ND031000
-B	. SP110000	Å LA280000	, SP080000	Ä LA180000	» SP180000	º SM200000	¿ SP160000	| SM130000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	Ö LO180000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	¦ SM650000	~ SD190000	@ SM050000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	! SP020000	^ SD150000	? SP150000	" SP040000	± SA020000	] SM080000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

Code Page 00278

Code Page 37, USA/Canada

HEX DIGITS 1ST → 2ND ↓	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	(SP) SP010000	& SM030000	- SP100000	ø LO610000	Ø LO620000	° SM190000	µ SM170000	^ SD150000	{ SM110000	} SM140000	\ SM070000	0 ND100000
-1	(RSP) SP300000	é LE110000	/ SP120000	É LE120000	a LA010000	j LJ010000	~ SD190000	£ SC020000	A LA020000	J LJ020000	÷ SA060000	1 ND010000
-2	â LA150000	ê LE150000	Â LA160000	Ê LE160000	b LB010000	k LK010000	s LS010000	¥ SC030000	B LB020000	K LK020000	S LS020000	2 ND020000
-3	ä LA170000	ë LE170000	Ä LA180000	Ë LE180000	c LC010000	l LL010000	t LT010000	· SD630000	C LC020000	L LL020000	T LT020000	3 ND030000
-4	à LA130000	è LE130000	À LA140000	È LE140000	d LD010000	m LM010000	u LU010000	© SM520000	D LD020000	M LM020000	U LU020000	4 ND040000
-5	á LA110000	í LI110000	Á LA120000	Í LI120000	e LE010000	n LN010000	v LV010000	§ SM240000	E LE020000	N LN020000	V LV020000	5 ND050000
-6	ã LA190000	î LI150000	Ã LA200000	Î LI160000	f LF010000	o LO010000	w LW010000	¶ SM250000	F LF020000	O LO020000	W LW020000	6 ND060000
-7	å LA270000	ï LI170000	Å LA280000	Ï LI180000	g LG010000	p LP010000	x LX010000	¼ NF040000	G LG020000	P LP020000	X LX020000	7 ND070000
-8	ç LC410000	ì LI130000	Ç LC420000	Ì LI140000	h LH010000	q LQ010000	y LY010000	½ NF010000	H LH020000	Q LQ020000	Y LY020000	8 ND080000
-9	ñ LN190000	ß LS610000	Ñ LN200000	` SD130000	i LI010000	r LR010000	z LZ010000	¾ NF050000	I LI020000	R LR020000	Z LZ020000	9 ND090000
-A	¢ SC040000	! SP020000	¦ SM650000	: SP130000	« SP170000	ª SM210000	¡ SP030000	[SM060000	(SHY) SP320000	1 ND011000	2 ND021000	3 ND031000
-B	. SP110000	$ SC030000	, SP080000	# SM010000	» SP180000	º SM200000	¿ SP160000	] SM080000	ô LO150000	û LU150000	Ô LO160000	Û LU160000
-C	< SA030000	* SM040000	% SM020000	@ SM050000	ð LD630000	æ LA510000	Ð LD620000	¯ SM150000	ö LO170000	ü LU170000	Ö LO180000	Ü LU180000
-D	(SP060000	) SP070000	_ SP090000	' SP050000	ý LY110000	¸ SD410000	Ý LY120000	¨ SD170000	ò LO130000	ù LU130000	Ò LO140000	Ù LU140000
-E	+ SA010000	; SP140000	> SA050000	= SA040000	þ LT630000	Æ LA520000	Þ LT640000	´ SD110000	ó LO110000	ú LU110000	Ó LO120000	Ú LU120000
-F	| SM130000	¬ SM660000	? SP150000	" SP040000	± SA020000	¤ SC010000	® SM530000	× SA070000	õ LO190000	ÿ LY170000	Õ LO200000	(EO)

Code Page 00037

IBM's codepage documentation can be found here:

http://www-03.ibm.com/systems/i/software/globalization/codepages.html

Appendix D – TN5250 Auto-signon

Telnet supports the capability for a user to bypass the Sign On display by sending a user profile name and password with the Telnet session request. The 5250 host uses the setting for the QRMTSIGN (Remote sign-on) system value to determine how to handle requests for automatic sign-on. The table below shows the options. These options apply only when the Telnet request includes a user ID and password.

The password is sent in a encrypted form according to the document RFC2877 (5250 Telnet Enhancements).

How QRMTSIGN Works with Telnet	
*REJECT	Telnet sessions that request automatic sign-on are not allowed.
*VERIFY	If the user profile and password combination is valid, the Telnet session starts.
*SAMEPRF	If the user profile and password combination is valid, the Telnet session starts.
*FRCSIGNON	The system ignores the user profile and password. The user sees the Sign-On display.

You set the username and password in the configuration program. The parameters are:

```
5250 Emulation\TN5250 Enhancements\User Name
5250 Emulation\TN5250 Enhancements\Password
```

Appendix E – TN5250 Enhancement Settings

The setting for TN5250 Enhancements can be specified in an external file. This is especially useful when those parameters are different between individual terminals. For example, some installations use a unique device name for each terminal to be able to identify the sessions.

Put these settings in a text file named TN5250Enhancements.txt in the same folder as your configuration file. You can specify one or more of the settings in the following format:

```
User Name=<user name>
Password=<password>
Current Library=<library>
Initial Menu=<menu>
Program To Call=<program>
Device Name=<device name>
```

Appendix F – Command Line Parameters

Command line parameters can be passed to Freefloat Access when started from a shortcut or registry key.

Currently, two parameters are supported:

```
[-autostart] [configuration file]
```

The *autostart* parameter, when specified, activates the *Startup Delay Time* option in the configuration file. The delay can be used when Freefloat Access is started before the network connection is initialized.

For example, a possible command line for starting Freefloat Access could look like this:

"C:\Program Files\Freefloat\Freefloat Access\Win32\win32.exe -autostart C:\Program Files\Freefloat\Freefloat Access\Win32\terminal.cfg"

Appendix G – Version History

Version	Changes	Date
2.0.1	Major new release. Completely new script engine based on Lua. Key name syntax changed from <> to {} to be consistent with Freefloat Link*One.	2010-03-08
2.0.2	Fixed problem with barcode received from Link*One when input wasn't allowed.	2010-04-09
2.0.3	Fixed a problem with cursor positioning in 5250 save/restore screen.	2010-04-15
2.0.4		2010-05-05
2.0.5	Added lua function isKeyboardLocked()	2010-05-11
2.0.6	Fix for Intermec scanner	2010-06-02
2.0.7	Repack, bad licenser.exe	2010-06-29
2.0.8	the fix in 2.0.3 was somehow lost, re-implemented	2010-09-10
2.0.9		2010-10-12
2.0.10	Barcode Reader option activated for SC600/PHL7000	2010-11-05
2.0.11	Bugfix: 5250 field format MONOCASE didn't work	2010-12-09
2.0.12	Fix for runaway script error when pushing the Status line Keyboard Reset Area when in Ansi mode	2010-12-14
2.0.13	Bugfix: Denso BHT400 SEND key didn't work unless keyboard was in FUNC mode. Bugfix: Denso Radio not started.	2011-01-11
2.0.14	Added support for Datalogic Falcon 44xx Windows Mobile	2011-01-18
2.0.15	Support for using SSL (experimental, not released)	2011-03-16
2.0.16	Added lua functions app.isInputAllowed(), app.disconnect(), app.powerOff() and onInactivityTimeOut(). Support for display of Numeric/Alpha keyboard state on Motorola MC9090 (WM)	2011-05-06
2.0.17	Added support for DLoG XMT5 Windows CE 6.0	2011-07-08
2.0.18	Bugfix: licenser.exe could not find the macaddress of WLAN-cards on Vista & Windows 7	2011-08-16
2.0.19	Added support for Intermec CK3	2011-09-21
2.0.20	Fix:FldExit key did not work on Intermec CK3	2011-10-19
2.0.21	Support for Basic TN3270E with LU selection added.	2011-11-09
2.0.22	Fix for crash when empty LUname & TN3270E selected. Various fixes for SSL issues. Added support for Opticon H21.	2011-11-17

Version	Changes	Date
	Added support for Teklogix Workabout, XT10 and 8515. Added support for Datalogic Falcon X3 Windows Mobile.	
2.0.23	Added the possibility to select font.	2011-11-18
2.0.24	Support for keys ALT-F1 to ALT-F12. Changed the behaviour of file/exit & CLOSE button to include a session restart question.	2011-11-30
2.0.25	Better support for 3270 extended highlighting attribute Added support for Falcon X3 CE Added support for Opticon H25 Added support for keyboard state indicator on Opticon H25	2012-01-18 2012-01-24
2.0.26	Added support for Intermec CS40 Support for generic PocketPC 2002 changed to PocketPC 2003	2012-04-12
2.0.27	Better handling of 3270 colors and extended highlighting attributes.	Not released
2.0.28	Added handling of TN3270E error reject, new toggle "Force TN3270E mode". Added logging of licensing and Messagebox when not licensed, also shows DEMO on statusline instead of battery %. When having an exit password, and being on the reconnect/Exit screen and clicking the close "X" was not handled.	2012-06-13
2.0.28	Fixed exit password double entry. Fixed bug in 3270 screen attribute.	2012-06-26
2.0.28	Added config option: "Enable Show Not Attached"	2012-07-04
2.0.29	Added support for Denso BHT1100 Changed the config option: "Snap Keyboard" to mean "the keyboard is snapped and centered to the bottom of the emulator window"	2012-07-31
2.0.30	Added support for Casio DT-X8 fixed Casio DT-X7 installation Fixed keytables.lua with entries for KEYAPP1-KEYAPP6	2012-08-14
2.0.31	Added lua function app.setBufferText()	2012-09-06
2.0.32	Fixed the lua function app.isInputAllowed() to better reflect the keyboard lock state. Bugfix: when using SSL the script function app.sleep() did not process keys and network as intended.	2012-09-19 2012-10-10
2.0.33	Bugfix: 5250 numeric field, minus character could not be	2012-10-16

Version	Changes	Date
	entered.	
2.0.34	Bugfix: 3270 Extended highlightning was not reset by the next start field.	2012-10-24
2.5	Changed product name to Freefloat Access	2012-11-01